A MULTI–LEVEL SECURE OBJECT-ORIENTED DATABASE MODEL

George B. Durham[1,2] Konstantinos Kalpakis[3]
Computer Science and Electrical Engineering Department
University of Maryland Baltimore County
1000 Hilltop Circle
Baltimore MD 21250

Abstract

This model presents a multi-level secure (MLS) database using object-oriented technology. The model is based on, and extends the requirements of the Department of Defense 5200.28-STD, DoD Trusted Computer System Evaluation Criteria (TCSEC) dated December 1985, commonly known as the Orange Book.

Currently, there does not exist a database model in any technology which meets the requirements of the Orange Book. There has been little interest outside of the U.S. Government and the academic community because the Orange Book is believed to focus on military needs rather than commercial needs. Since commercial espionage is growing daily, and without proper protection, commercial information will be pilfered both nationally and internationally, our model is of interest to commercial users as well.

Previous work has focused on Discretionary Access Controls (DAC), Mandatory Access Controls (MAC), or other security requirements not included in the Orange Book, but no work includes all three.

We develop policies for access controls, inference controls, and implementation strategy based on the MAC, DAC, and other security requirements. The access authorization mechanism is based on a combination of DAC and MAC requirements, and the proposed model is easily extended to include other access requirements. We also present an outline for implementing our model.

Keywords: Database security, object–oriented databases, mandatory access controls, discretionary access controls, inference channels, multi–level security.

I. Introduction

Many organizations today want to distribute information via the Internet and World Wide Web, yet they also have files to which they want to restrict access. Even within an organization, some information may be distributed to all employees, while other information need go only to select groups of employees. These organizations must employ a Multi-Level Secure (MLS) database to provide for variable access controls, yet have efficient access to information.

[1]Supported in part by the Department of Defense Advanced Studies Program.
[2]Email: gdurha1@cs.umbc.edu
[3]Endorsing Advisor

Although each organization will have its own set of security requirements based on its organization and purpose, there are certain general policies and requirements which must exist. Because there is no agreement on, or definition of a secure database model outside of the U.S. Government, this paper will emphasize the requirements as set forth in the "Department of Defense Trusted Computer System Evaluation Criteria" (TCSEC), commonly known as the Orange Book [5].

The model described in this paper ties together many of the concepts developed in earlier works, along with the Orange Book requirements, to form a base for the model. It then refines and extends the basic policies and concepts to present a composite MLS database model which meets the needs of the military and commercial customers. [4] Such a model has not previously been defined, since most research has emphasized only one aspect of the security requirements, rather than looking at all of them simultaneously.

The remainder of this paper is organized as follows: In section II, we present a brief overview of the TCSEC divisions and classes, define some of the concepts of secure databases and information security, and discuss previous work. In section III, we develop the security policies and implementation strategies for our model. In section IV, we develop the database model. In section V, we explain the implementation of the security policies and implementation strategies. In section VI, we present some conclusions about our model, and suggest areas for future work.

II. Preliminaries and Previous Work

In this section, we discuss security policies and concepts that are essential to our model. Further, we review previous work in database security, related to our model.

II-1. DOD 5200-28-STD: The Orange Book

The requirements for this model are based on the TCSEC [5], which describes four divisions of trust for computer systems. The four divisions, D through A, with A being the highest level of assurance, have subdivisions, known as classes. Each division and subdivision within the TCSEC is defined by its requirements for implementation and validation of either Discretionary Access Control (DAC) or Mandatory Access Control (MAC), or both. In general, DAC may be thought of as restricting a user's access to information based on their position or membership in a group, and thus, is the user's "need-to-know" the information in order to fulfill their duties. MAC may be thought of as restricting a user's access to information based on that user's security clearance level as compared to the assigned security level of the information. The interested reader is referred to the source [5] for further detail.

II-2. Preliminaries

We define a secure database as one which can be shown to authorize or deny user access to information in accordance with the security policies of an organization. The set of security policies covers three areas: security clearance, security level, and need-to-know rules. A security clearance is a measure of the level of trust given to an individual. The security level is a label assigned to information, designating the information's classification level as

[4] The model assumes a Trusted Computing Base (TCB) which is invulnerable to outside attack.

one of a set of hierarchical classification labels, and possibly, one of a set of non-hierarchical compartment labels. Need-to-know rules are dynamic rules which define the information a person needs to know in order to do their job.

We also need to introduce the idea of access controls. Abrams and Smith [1] describe access control policies for databases in reference to a Generalized Framework for Access Control (GFAC), which contains three components: Access Control Information (ACI), Access Control Rules (ACR), and Access Control Authority (ACA). Abrams and Smith [1] believe that the traditional MAC and DAC requirements are too restrictive to cover the broad spectrum of access control policies, and thus, the system must be able to automatically change data classification labels as the values or system attributes change.

II-3. Previous Work

Bell and Lapadula [2] formally define a computer security model which is the defacto standard for most later work, and include the *-property, which states that a subject, which is an active entity, may access an object, which is a passive entity, only if the subject's security level is equal to or greater than the security level of the object, and any non-hierarchical categories of the object are included in the subject's categories. When a subject represents a user, the subject's security level equates to the security clearance of the user. The TCSEC [5], which relies heavily on the work of Bell and Lapadula [2], was the first formal definition for Multi-Level Secure (MLS) computing. Denning [4] provides an introduction to covert channels and inference control theories, which need to be considered when evaluating a secure computing system.

Several research efforts have focused on DAC. Fernandez *et al.* [7, 8, 6] use formal models to describe access authorization in the form of directed graphs based on inherited permissions between superclasses and their subclasses. Gudes *et al.* [10] extend the above model to include negative authorization. Kelter's model [13, 14], based on access control lists (ACL), provides a finer level of granularity, and models access control of types as well as objects. The user-role based model of Demurjian *et al.* [3] provides a thorough presentation of authorization analysis.

Other efforts have focused on MAC requirements. Garvey and Lunt [9] provide a unique approach to an MLS database with a study of using knowledge base rules as access controls to an object-oriented database. Jajodia and Kogan [11], Sandhu *et al.* [22], and Schaefer *et al.* [23] base their studies on a Trusted Computer Base (TCB) filter controlling access to the database. The uniqueness of Schaefer *et al.* [23] is that it has been implemented as a prototype. McCollum *et al.* [17] focus on restricting access by allowing the originator to control access authorization via an ACL, which is attached to each object.

Millen and Lunt [18] focus on the inference problem. Their model does not allow a message to directly invoke the method of an object, but rather, a new subject is created at the object for the sole purpose of invoking the proper method on behalf of the message. Marks *et al.* [16] and Motro *et al.* [19] present a scheme for solving the inference problem in relational databases via a user access history list and pre-defined thresholds of aggregation.

Keefe and Tsai [12] provide a comparison of three object-oriented models against what the authors define as general principles for "well formed" multilevel databases. Their contribution is that they show the strengths and weakness of the three models, and offer insight

into possibilities for a joint DAC/MAC model. Thuraisingham [25] discusses a multilevel multimedia database system which reassembles a complex object from its many parts at a requested level. Sibley and Smith [24] provide a study of secure behavior in which the system response to incorrect actions varies depending on the policy, or interpretation of policy of the particular implementation.

III. Policy Definitions and Descriptions

We provide security definitions first, then policies, and finally strategies for implementation which will help ensure the security of our model.

III-1. Definitions

Level is defined to be any one of a set of security levels which form a linear order $L = \{l_1, l_2, \ldots, : l_1 \prec l_2\}$, from lowest to highest, and that conform to the requirements set forth for MAC in the TCSEC.

Access-Requirement $\mathrm{AR} = \{\mathrm{ar}_1, \mathrm{ar}_2, \ldots\}$ is defined to be any one of a set of labels which denote the accessibility to data elements required to accomplish a defined task. The set Access-Requirements is an arbitrary set of labels whose composition varies over time and can be expressed through DAC as set forth in the TCSEC.

Tasking $T = \{\ t = (l, \mathrm{ar}) \ : \ l \in L \ \text{ and } \ \mathrm{ar} \in \mathrm{AR}\ \}$ is defined to be a pair of Level and Access-Requirement such that it delimits the access at a particular security level which is needed to fulfill a defined job function. We show the mapping of users and objects to Taskings in section IV-2.

III-2. Basic Security Policies

BP1 Data boundaries depend on Tasking. Access to data is limited by both Level and Access-Requirement, the combination of which forms the limitations required for levels of security access, and implements the policy of least privilege or need to know, thus complying with both DAC and MAC requirements.

BP2 Write authorization is only for the current Tasking. A user can write only to their current Level and within their current Access-Requirement. They must exit the database and re-enter in the appropriate Tasking in order to write to a higher or lower Level, or to another Access-Requirement at their current Level.

BP3 Read authorization is for the current Tasking and any authorized lower level objects which are in the Access-Requirement defined for the current Tasking. Read up is not allowed.

BP4 A user is granted only one database access at a time. This policy ensures that there is no data flow between two concurrent processes of a single user.

BP5 Administration is a Tasking. Administrative duties are defined in Taskings which allows for standardization of user access, and also limits administrators with the need to know requirement.

BP6 There is no automatically inherited access authorization between classes. To be authorized access, a Tasking must have explicit authorization to each object.

BP7 The granularity of access authorization is variable; authorization may be to an entire class, object, or attribute. This policy allows for the flexibility needed to define Taskings which will exactly meet the needs of the users, but not be in violation of either MAC or DAC.

III-3. Implementation Strategies

The following strategies extend the basic policies to better refine the security mechanisms of the model.

IS1 Tasking authorization is verified for each action. The system must ensure that no user can access data based on an outdated authorization, and therefore, must take action to revalidate all users as soon as any user access is modified.

IS2 Full authorization to all objects needed to execute a query is required. This policy limits inference channels.

IS3 All database request failures, with limited exception, return the same information to the user. This policy eliminates an inference channel. The limited exceptions are system failures which are independent of the requested action; e.g., an operating system failure.

IS4 Reuse of a user object identifier from a lower Level, or a different Access-Requirement will succeed. This policy results in allowable polyinstantiation at the expense of database complexity. It is necessary to block information inference.

IS5 There will not be multiple copies of data items within the database. Access to a data item from multiple Taskings will reference the same actual data item.

IS6 Once an object is created within the database, it will remain at its creation Level and Access-Requirement for the life of the database unless all active references are deleted. When a user in one Tasking deletes an object, that object continues to exist for all other Taskings which reference it.

IS7 No object may be automatically reclassified. To modify the Level of an object, that object must be deleted from the database and re-created at its new Level.

IV. Database Model

The database model provides a formal description in terms of the data as objects, and the functions to access the data as methods. The database security is implemented as access control functions.

IV-1. Database Objects

The term "object-oriented" evokes different thoughts for different people. The terms associated with object technology are given varying interpretations depending on the author and circumstances. Rishe [20], Rumbaugh *et al.* [21], and Kim and Lochovsky [15] provide further explanation of object-oriented concepts and design interpretations. By strict definition, all elements in an object-oriented system are objects. Jajodia and Kogan [11] use this type of definition in their statement that their model has only objects, and no subjects. We,

however, will use the term object in the sense that an object is a passive element stored in the database. A subject is an active element, usually invoked on behalf of a user, which sends a message to an object to activate a method for that object.

A database is a triple DB $= (U, S, O)$, where $U = \{\ (u_1, \mathrm{st}_1), (u_2, \mathrm{st}_2), \ldots\ :\ u_i$ is either a single user or a group of users, and st_i is the user status of active or inactive $\}$ (users), $S = \{s_1, s_2, \ldots\}$ (subjects), and $O = \{o_1, o_2, \ldots\}$ (objects).

An object Obj = (OID, Val, M) has an identifier, a value, and a set of methods. The identifier (OID), is a system assigned unique label. The value (Val) may be an atomic value, or it may be a complex value of multiple objects. Methods (M) is a non-empty set of methods used to access the values in Val.

There is a set User_ID containing tuples (UOID, Tasking). The UOIDs are user generated labels which are human readable, and the user uses to reference the objects. The UOID must be paired with a Tasking because one UOID may reference multiple objects from different Access-Requirements. We define **map_object**(uoid,t) to be a function which given a UOID and Tasking returns the OID of the associated object. There may be a 1–to–1 or a 1–to–many relationship between UOID and OID. There may not be a many–to–1 or a many–to–many relationship.

Objects are instantiations of classes, thus, each object is a member of a single class which is defined for the database.

Our object-oriented database schema is a rooted directed acyclic graph. At the root, there is a class, known as a metaclass, which is the superclass of all other classes. There is a subclass–superclass relationship $\lhd$, between the elements of the set of classes $C = \{c_1, c_2, \ldots\}$ in the database, which forms a partial order. We write $c_i \lhd c_j$ whenever c_i is a subclass of c_j. Therefore, other classes are either subclasses of the metaclass, or they are subclasses of other subclasses of the metaclass.

IV-2. Access Verification

Verification of a user to access data in the database is a 2–step process. After system login, to gain initial entry into the database, user verification **Vu**(u, t) must occur for the user in a selected Tasking. Let **status**(u) be a function which given a user u, returns true if the user is not active in the database, and false otherwise. Also, let **member**(u, t) be a function which given a user u and a Tasking t, returns true if the user is defined for the Tasking, and false otherwise. We define **Vu**(u, t) to return null, if the user or Tasking are invalid; accepted if **status**(u) and **member**(u, t) are true; and, rejected otherwise.

The second step is an operation verification **Vop**(uoid,m, t) which occurs whenever the user makes a request to the database. The user request is parsed so that current authorization to each object needed to satisfy the request is verified. Let **map_object**(uoid,t) be a function which takes a UOID uoid and a Tasking t, and returns the OID of the referenced object. Let **allowed**(o, m, t) be a function which takes an object o, a mode m, and a Tasking t, and returns a method if the Tasking t is defined for mode m on object o; null otherwise. Mode is defined to be any one of the set Mode = {Read, Update, Create, Delete}. Operation verification is defined as returning null if the object o or mode m are not valid; a method if **allowed**(o, m, t) returns a method; and, rejected otherwise.

IV-3. Reference Structure

An important feature of an MLS database is the capability for a user to have read access to the values of objects which are below the user's level as well as access to object values at the user's level. Our model uses a Multiple Object Block (MOB) to implement that capability. The MOB is a dynamically created list of references to objects, indexed in some predefined order. In our model, we define the order to have the "primary reference" at the beginning of this list. The other objects in this list are in descending order of their associated Levels, cover stories first, then re-created objects.

Two associated mechanisms are the Multiple Object Prompt (MOP) and the Deleted Object Prompt (DOP). The MOP prompts the user whenever the system finds an MOB associated with a tuple (uoid,t) to enable the user to designate which of the multiple referenced objects is desired. The DOP notifies a user that a lower Level user has deleted the object, from his or her space, which the current user is requesting. The DOP allows the user to confirm the delete, request the object to persist, or re-create the object at the current user's Level.

Take for example a 1-to-1 relationship between UOID and OID where the UOID is defined for multiple Taskings. In this circumstance, there are multiple references from the UOID list to an OID, which references an object, as in Figure 1.

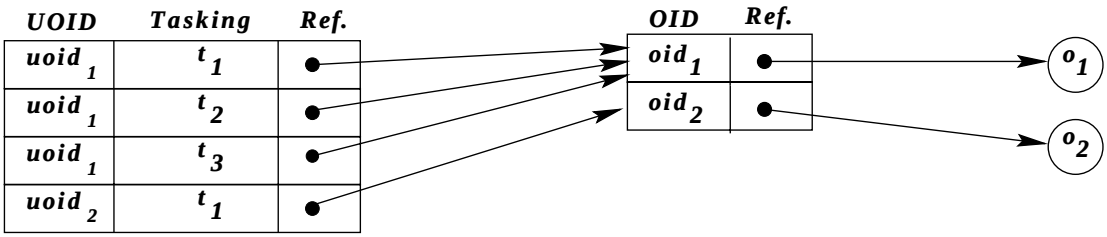

Figure 1: 1–to–1 Relationship

The state represented by Figure 1 is that where Tasking t_1, t_2, and t_3 all have the same Access-Requirement, but security levels unclassified, confidential, and secret, respectively, for example. Objects o_1 and o_2, therefore, represent objects at the unclassified Level.

Now, let a user in Tasking t_1 delete the object with UOID uoid_1, and then create a new object using uoid_1 as the UOID. At the same time, the users in t_2 and t_3 may want object o_1 to persist. This state is accomplished with the use of a MOB, with a list of references to multiple objects. The system creates the MOB when it recognizes the existence of multiple objects being referenced from within a common Access-Requirement. The result is shown in Figure 2.

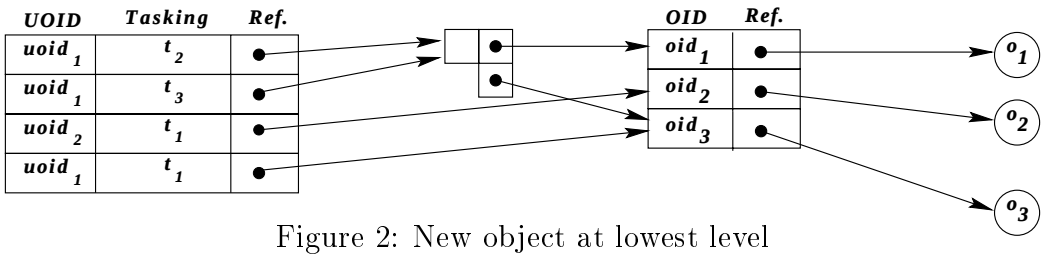

Figure 2: New object at lowest level

The assumption we make here is that until a user deletes an existing object, that object

remains of primary interest to the user over any newly created object at a lower level.

Going back to the original state as in Figure 1, let a user in Tasking t_2 delete his or her reference to o_1, then create a new object using uoid_1. Cover stories can be created in this manner. The complexity of this create arises from the fact that although this action will not affect t_1's view, it will affect t_3's view since both o_1, and the newly created object will be within t_3's view, and for that matter, o_1 remains within t_2's view. An alternative would be to only allow the user to view the nearest object, and require the user to login at the appropriate level to see a cover story object. The security aspects are unaffected by the choice of implementation. Figure 3 shows the result.

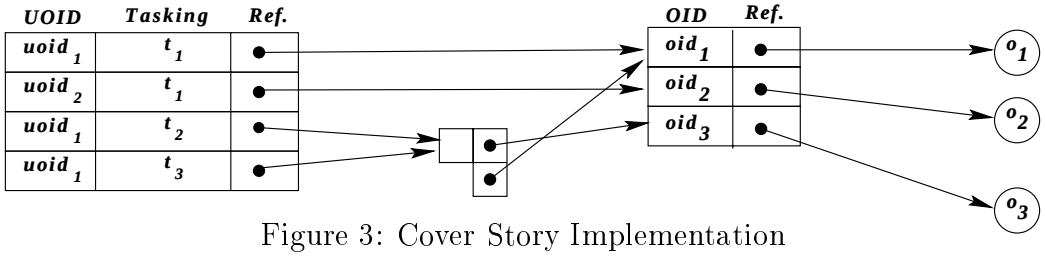

Figure 3: Cover Story Implementation

We make the assumption that a user at any Level will be primarily interested in viewing the nearest object, that is, an object at the same, or next lower Level within that user's view. Therefore, the primary object which is accessed by t_3 through UOID uoid_1 will be o_3, not o_1.

V. Outline for Implementing Policies in the Model

We now discuss the security requirements in terms of the access controls to show that the model implements the requirements as given in section III.

V-1. Basic Policy Implementation

BP1, **BP2**, **BP3**, and **BP5** rely on **allowed**(o, m, t) returning the correct method. If an incorrect method is returned, then either **allowed**(o, m, t) is in error, or one or more of the other mapping functions has an error in them, and the incorrect parameters were passed to **allowed**. **BP4** relies totally on **status**(u) to provide a correct mapping for the specified user. **BP6** is accomplished during class definition along with **BP7**. In both cases, the class methods must be properly defined such that they will be accessible only when Taskings are defined for them, and in the case of **BP7**, that they properly access methods of other objects.

V-2. Implementation Strategy Design

IS1 and **IS2** rely on correct mappings within verification operation such that a correct, current method is returned for each object in the user request. **IS3** is accomplished with exception handlers. **IS4** and **IS5** both require the correct implementation of MOBs. **IS6** and **IS7** rely on the MOBs as well, but they also require proper action in response to DOPs.

VI. Conclusions

When implemented as previously described, this model provides an MLS database meeting the stated requirements. Although there is no aggregate policy stated, the necessity of one is determined by the implementation. Also, some organizations may not require

such a high level of assurance for their particular use. Variations of the model can easily be implemented because of the underlying object-oriented technology. The object access assumptions made in this model can easily be changed for implementations with other requirements. There may not be a requirement, for instance, that a user be able to read the value of all objects at lower levels referenced by a UOID within an Access-Requirement. The user may only require access to the nearest object.

Other models [11, 10] allow inheritance of access, and use different access control mechanisms, such as negative authorization, and class security hierarchy rules to limit access via inheritance. We believe that our model, which does not allow inheritance of access, is far less complex, and has access controls which are more verifiable, and therefore is superior in its implementation.

Our next step is to build a prototype demonstrating the feasibility and properties of our model. We also plan to incorporate aggregate policies in our model. Much of the basis of such policies depends on semantic interpretations being developed for the data.

The security policies of this model are designed to meet the needs of the U.S. government. To make our model more attractive to commercial users, one might relax some of the security requirements, and thereby produce a more user friendly and simpler system.

Acknowledgments

I acknowledge with deep gratitude the technical guidance, advice, and critique provided by my Advisor, Dr. Konstantinos Kalpakis, the moral support of Dr. Yelena Yesha, and the guidance of Dr. Charles Abzug. I would also like to thank my co-worker, Mr. Donald Marks, for the many discussions on security policies and interpretations.

References

[1] M. D. Abrams and G. W. Smith. A trusted basis for database access controls. Technical report, The Mitre Corporation, McLean, VA, and National Defense University, Washington, D.C., 1991.

[2] D.E. Bell and L.J. LaPadula. Secure computer systems: Unified exposition and multics interpretation. Technical report, The Mitre Corporation, Bedford, MA, 1976.

[3] S.A. Demurjian, M.Y. Hu, T.C. Ting, and D. Kleinman. Towards an authorization mechanism for user-role based security in an object-oriented design model. In *Proceedings of the 12th Annual International Phoenix Conference on Computers and Communications*, pages 195–202, 1993.

[4] D. E. Denning. *Cryptography and Data Security.* Addison-Wesley Publishing Company, Reading, MA, 1982.

[5] DoD. Trusted computer system evaluation criteria. Technical Report DoD 5200.28-STD, Department of Defense, National Computer Security Center., 1985.

[6] E. B. Fernandez, R.B. France, and D. Wei. A formal specification of an authorization model for object-oriented databases. In *Proceedings of the IFIP WG 11.3 Ninth Annual Working Conference on Database Security*, pages 105–119, 1995.

[7] E. B. Fernandez, E. Gudes, and H. Song. A security model for object-oriented databases. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pages 110–115, 1989.

[8] E. B. Fernandez, E. Gudes, and H. Song. A model for evaluation and administration of security in object-oriented databases. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):275–292, 1994.

[9] T. D. Garvey and T. F. Lunt. Multilevel security for knowledge based systems. Technical Report SRI-CSL-91-01, SRI International, Menlo Park, CA, 1991.

[10] E. Gudes, H. Song, and E. B. Fernandez. Evaluation of negative, predicate, and instance-based authorization in object-oriented databases. Technical report, Ben-Gurion University, Beer-Sheva, Israel, and Florida Atlantic University, Boca Raton, FL., 1991.

[11] S. Jajodia and B. Kogan. Integrating an object-oriented data model with multilevel security. In *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pages 76–85, 1990.

[12] T. F. Keefe and W. T. Tsai. Security model consistency in secure object-oriented systems. In *Proceedings of the Fifth Annual Computer Security Applications Conference*, pages 290–298, 1989.

[13] U. Kelter. Group-oriented discretionary access controls for distributed structurally object-oriented database systems. In *Proceedings of the European Symposium on Research on Computer Security*, pages 23–33, 1990.

[14] U. Kelter. Discretionary access controls in a high-performance object management system. Technical report, Fern Universitat Hagen, Praktische Informatik V, Hagen, Germany, 1991.

[15] W. Kim and F. H. Lochovsky. *Object-Oriented Concepts, Databases and Applications.* ACM Press, New York, NY, 1989.

[16] D. G. Marks, A. Motro, and S. Jajodia. Enhancing the controlled disclosure of sensitive information. In *Proceedings of Computer Security - ESORICS 96, 4th European Symposium on Research in Computer Security*, pages 290–303, 1996.

[17] C. J. McCollum, J. R. Messing, and L. Notargiacomo. Beyond the pale of MAC and DAC - defining new forms of access control. In *Proceedings IEEE Symposium on Security and Privacy*, 1990.

[18] J. K. Millen and T. F. Lunt. Security for object-oriented database systems. In *Proceedings of the 1992 IEEE Computer Society Symposium on Security and Privacy*, pages 260–272, 1992.

[19] A. Motro, D. G. Marks, and S. Jajodia. Aggregation in relational databases: Controlled disclosure of sensitive information. In *Proceedings of ESORICS-94, Third European Symposium on Research in Computer Security*, pages 431–445, 1994.

[20] N. Rishe. *Database Design.* McGraw-Hill, Inc., New York, NY, 1992.

[21] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1991.

[22] R. Sandhu, R. Thomas, and S. Jajodia. A secure kernelized architecture for multilevel object-oriented databases. In *Proceedings of the IEEE Computer Security Foundations Workshop IV*, pages 139–152, 1991.

[23] M. Schaefer, P. Martel, T. Kanawati, and V. Lyons. Multilevel data model for the Trusted ONTOS Prototype. In *Proceedings of the IFIP WG 11.3 Ninth Annual Working Conference on Database Security*, pages 121–141, 1995.

[24] E. H. Sibley and G. W. Smith. On the behavior of multilevel database systems. Technical report, George Mason University and National Defense University, 1991.

[25] B. Thuraisingham. Multilevel security for multimedia database systems. Technical report, The Mitre Corporation, Bedford, MA., 1991.